Color Adaptive Watermarking

Related Application Data

This application is a continuation of US Patent Application No. 09/553,084 filed April 19, 2000 which is a continuation in part of US Patent Application 09/503,881 filed February 14, 2000, which are each hereby incorporated by reference.

Technical Field

The invention relates to image processing and specifically relates to color image processing.

Background and Summary

In color image processing applications, it is useful to understand how humans perceive colors. By understanding the human visual system and its sensitivity to certain colors, one can more effectively create and manipulate images to create a desired visual effect. This assertion is particularly true in image processing applications that intentionally alter an image to perform a desired function, like hiding information in an image or compressing an image. In digital watermarking, for example, one objective is to encode auxiliary information into a signal, such as an image or video sequence, so that the auxiliary information is substantially imperceptible to humans in an output form of the signal. Similarly, image compression applications seek to decrease the amount of data required to represent an image without introducing noticeable artifacts.

The invention relates to selective color masking of images. One aspect of the invention is a method for mapping a change in an image attribute such as luminance or chrominance to a change in color components such that the change is less visible. This form of color masking is particularly useful in watermark encoding applications, but applies to other applications as well.

A color masking method may be incorporated into a watermark encoder to reduce visibility of a watermark. The watermark encodes auxiliary information in an image by modifying attributes of image samples in the image. As part of the encoding process, the encoder computes a change in an image attribute, such as luminance, to an image sample to

encode auxiliary information in the image. It changes color values of the image sample to effect the change in luminance with minimized impact on visibility.

In one implementation, a mapping process transforms a change in an image sample attribute, such as luminance, to a change in color components of the image sample based on the color values of the image sample. The mapping process may be implemented with a look up table, where the image sample color values are used to look up a corresponding change in color values. For images represented as an array of color vectors (e.g., color triplets like Red Green Blue or Cyan Magenta Yellow), the look up table may be implemented as a multidimensional look up table with color vectors used to index a corresponding change in the color values of the image samples. The mapping process may transform a change in an image sample attribute to an absolute change in the color values of the sample, or to a relative change, such as a scale factor. To effect a change in an image sample, for example, the mapping process uses the color of an image sample to look up a scale factor for each of the color values of the image sample. It uses this scale factor, along with the desired change in the image attribute, to compute an absolute change in the color values.

Another aspect of the invention is a color masking method that may be used in digital watermark and other applications. The method reads the color values of an image sample and a corresponding change of an attribute of the image sample. Based on the color values of the image sample, the method maps the change in the image sample attribute to a change in color components of the image sample. The change in the color components is equivalent to the change in the image sample attribute, yet reduces visibility of the change for the specific color values of the image sample. The image sample attribute may be luminance, chrominance, or some other attribute.

Another aspect of the invention is the use of a color key to determine validity of a watermark in an image. The color key indicates how an attribute of image sample is changed to encode a watermark based on the color of that image sample. Since a watermark encoded using this color key is dependent on the color key and the specific color values of the image in which it is embedded, the color key can be used to check whether a watermark is valid in an image suspected of containing a watermark. For example, if the watermark is forged without the knowledge of the key or copied from a different image, then the decoder is likely

to determine that the watermark has not been encoded in the color channels specified by the color key. An invalid watermark will likely be weak in the color channels specified by the key. As such, invalid watermarks found in the wrong color channels or having a weak signal strength in the correct color channels as indicated by the key are considered to be invalid.

Another aspect of the invention is a user interface method that enables a user to control the strength of a watermark in specified colors or color regions of an image. The user interface enables the user to specify the color or color regions to embed a watermark signal more or less strongly than other colors. The transition into selected color regions is made less visible, by smoothly changing the signal strength depending on the distance from the selected color region. Also, it enables the user to select the color region by selecting pixels having the desired color in the image to be watermarked.

The color masking technology outlined above applies to image processing operations performed in the spatial domain as well as in other transform domains. For example, another aspect of the invention is a method that performs color adaptive encoding of auxiliary data in transform coefficients of the image. In this method, image blocks are transformed into transform coefficients, which are then altered to encode auxiliary data. The alterations are adaptive to the color of the image because they are dependent on the characteristic color of the block to which they are made. In one implementation, for example, the average color of the block is used to look up the corresponding color channels in which to embed the watermark.

Further features of the invention will become apparent with reference to the following detailed description and accompanying drawings.

Brief Description of the Drawings

Fig. 1 is a block diagram illustrating a watermark system.

Fig. 2 is a block diagram illustrating a watermark embedder.

Fig. 3 is a spatial frequency domain plot of a watermark orientation signal.

Fig. 4 is a flow diagram of a process for detecting a watermark signal in an image and computing its orientation within the image.

Fig. 5 is a flow diagram of a process reading a message encoded in a watermark.

Fig. 6 is a diagram depicting an example of a watermark detection process.

Fig. 7 is a diagram depicting the orientation of a transformed image superimposed over the original orientation of the image at the time of watermark encoding.

Fig. 8 is a diagram illustrating an implementation of a watermark embedder.

Fig. 9 is a diagram depicting an assignment map used to map raw bits in a message to locations within a host image.

Fig. 10 illustrates an example of a watermark orientation signal in a spatial frequency domain.

Fig. 11 illustrates the orientation signal shown in Fig. 10 in the spatial domain.

Fig. 12 is a diagram illustrating an overview of a watermark detector implementation.

Fig. 13 is a diagram illustrating an implementation of the detector pre-processor depicted generally in Fig. 12.

Fig. 14 is a diagram illustrating a process for estimating rotation and scale vectors of a watermark orientation signal.

Fig. 15 is a diagram illustrating a process for refining the rotation and scale vectors, and for estimating differential scale parameters of the watermark orientation signal.

Fig. 16 is a diagram illustrating a process for aggregating evidence of the orientation signal and orientation parameter candidates from two or more frames.

Fig. 17 is a diagram illustrating a process for estimating translation parameters of the watermark orientation signal.

Fig. 18 is a diagram illustrating a process for refining orientation parameters using known message bits in the watermark message.

Fig. 19 is a diagram illustrating a process for reading a watermark message from an image, after re-orienting the image data using an orientation vector.

Fig. 20 is a diagram of a computer system that serves as an operating environment for software implementations of a watermark embedder, detector and reader.

Fig. 21 is a diagram illustrating a process for mapping changes in image sample attributes to changes in color values of the image samples to minimize visibility of the changes.

Fig. 22 is a diagram of a color space depicting how to scale a color vector to black to effect a change in luminance.

Fig. 23 is a diagram of a color space depicting how to scale a color vector to white to effect a change in luminance.

Fig. 24 is a diagram of a color space depicting how to scale a color vector to effect a change in chrominance.

Fig. 25 is a diagram of a color space illustrating a method for enabling a user to control watermark strength based on colors.

Detailed Description

1.0 Introduction

A watermark can be viewed as an information signal that is embedded in a host signal, such as an image, audio, or some other media content. Watermarking systems based on the following detailed description may include the following components: 1) An embedder that inserts a watermark signal in the host signal to form a combined signal; 2) A detector that determines the presence and orientation of a watermark in a potentially corrupted version of the combined signal; and 3) A reader that extracts a watermark message from the combined signal. In some implementations, the detector and reader are combined.

The structure and complexity of the watermark signal can vary significantly, depending on the application. For example, the watermark may be comprised of one or more signal components, each defined in the same or different domains. Each component may perform one or more functions. Two primary functions include acting as an identifier to facilitate detection and acting as an information carrier to convey a message. In addition, components may be located in different spatial or temporal portions of the host signal, and may carry the same or different messages.

The host signal can vary as well. The host is typically some form of multi-dimensional media signal, such as an image, audio sequence or video sequence. In the digital domain, each of these media types is represented as a multi-dimensional array of discrete samples. For example, a color image has spatial dimensions (e.g., its horizontal and vertical components), and color space dimensions (e.g., YUV or RGB). Some signals, like video,

have spatial and temporal dimensions. Depending on the needs of a particular application, the embedder may insert a watermark signal that exists in one or more of these dimensions.

In the design of the watermark and its components, developers are faced with several design issues such as: the extent to which the mark is impervious to jamming and manipulation (either intentional or unintentional); the extent of imperceptibility; the quantity of information content; the extent to which the mark facilitates detection and recovery, and the extent to which the information content can be recovered accurately.

For certain applications, such as copy protection or authentication, the watermark should be difficult to tamper with or remove by those seeking to circumvent it. To be robust, the watermark must withstand routine manipulation, such as data compression, copying, linear transformation, flipping, inversion, etc., and intentional manipulation intended to remove the mark or make it undetectable. Some applications require the watermark signal to remain robust through digital to analog conversion (e.g., printing an image or playing music), and analog to digital conversion (e.g., scanning the image or digitally sampling the music). In some cases, it is beneficial for the watermarking technique to withstand repeated watermarking.

A variety of signal processing techniques may be applied to address some or all of these design considerations. One such technique is referred to as spreading. Sometimes categorized as a spread spectrum technique, spreading is a way to distribute a message into a number of components (chips), which together make up the entire message. Spreading makes the mark more impervious to jamming and manipulation, and makes it less perceptible.

Another category of signal processing technique is error correction and detection coding. Error correction coding is useful to reconstruct the message accurately from the watermark signal. Error detection coding enables the decoder to determine when the extracted message has an error.

Another signal processing technique that is useful in watermark coding is called scattering. Scattering is a method of distributing the message or its components among an array of locations in a particular transform domain, such as a spatial domain or a spatial

frequency domain. Like spreading, scattering makes the watermark less perceptible and more impervious to manipulation.

Yet another signal processing technique is gain control. Gain control is used to adjust the intensity of the watermark signal. The intensity of the signal impacts a number of aspects of watermark coding, including its perceptibility to the ordinary observer, and the ability to detect the mark and accurately recover the message from it.

Gain control can impact the various functions and components of the watermark differently. Thus, in some cases, it is useful to control the gain while taking into account its impact on the message and orientation functions of the watermark or its components. For example, in a watermark system described below, the embedder calculates a different gain for orientation and message components of an image watermark.

Another useful tool in watermark embedding and reading is perceptual analysis. Perceptual analysis refers generally to techniques for evaluating signal properties based on the extent to which those properties are (or are likely to be) perceptible to humans (e.g., listeners or viewers of the media content). A watermark embedder can take advantage of a Human Visual System (HVS) model to determine where to place an image watermark and how to control the intensity of the watermark so that chances of accurately recovering the watermark are enhanced, resistance to tampering is increased, and perceptibility of the watermark is reduced. Similarly, audio watermark embedder can take advantage of a Human Auditory System model to determine how to encode an audio watermark in an audio signal to reduce audibility. Such perceptual analysis can play an integral role in gain control because it helps indicate how the gain can be adjusted relative to the impact on the perceptibility of the mark. Perceptual analysis can also play an integral role in locating the watermark in a host signal. For example, one might design the embedder to hide a watermark in portions of a host signal that are more likely to mask the mark from human perception.

Various forms of statistical analyses may be performed on a signal to identify places to locate the watermark, and to identify places where to extract the watermark. For example, a statistical analysis can identify portions of a host image that have noise-like properties that are likely to make recovery of the watermark signal difficult. Similarly, statistical analyses may be used to characterize the host signal to determine where to locate the watermark.

Each of the techniques may be used alone, in various combinations, and in combination with other signal processing techniques.

In addition to selecting the appropriate signal processing techniques, the developer is faced with other design considerations. One consideration is the nature and format of the media content. In the case of digital images, for example, the image data is typically represented as an array of image samples. Color images are represented as an array of color vectors in a color space, such as RGB or YUV. The watermark may be embedded in one or more of the color components of an image. In some implementations, the embedder may transform the input image into a target color space, and then proceed with the embedding process in that color space.

2.0 Digital Watermark Embedder and Reader Overview

The following sections describe implementations of a watermark embedder and reader that operate on digital signals. The embedder encodes a message into a digital signal by modifying its sample values such that the message is imperceptible to the ordinary observer in output form. To extract the message, the reader captures a representation of the signal suspected of containing a watermark and then processes it to detect the watermark and decode the message.

Fig. 1 is a block diagram summarizing signal processing operations involved in embedding and reading a watermark. There are three primary inputs to the embedding process: the original, digitized signal 100, the message 102, and a series of control parameters 104. The control parameters may include one or more keys. One key or set of keys may be used to encrypt the message. Another key or set of keys may be used to control the generation of a watermark carrier signal or a mapping of information bits in the message to positions in a watermark information signal.

The carrier signal or mapping of the message to the host signal may be encrypted as well. Such encryption may increase security by varying the carrier or mapping for different components of the watermark or watermark message. Similarly, if the watermark or watermark message is redundantly encoded throughout the host signal, one or more encryption keys can be used to scramble the carrier or signal mapping for each instance of

the redundantly encoded watermark. This use of encryption provides one way to vary the encoding of each instance of the redundantly encoded message in the host signal. Other parameters may include control bits added to the message, and watermark signal attributes (e.g., orientation or other detection patterns) used to assist in the detection of the watermark.

Apart from encrypting or scrambling the carrier and mapping information, the embedder may apply different, and possibly unique carrier or mapping for different components of a message, for different messages, or from different watermarks or watermark components to be embedded in the host signal. For example, one watermark may be encoded in a block of samples with one carrier, while another, possibly different watermark, is encoded in a different block with a different carrier. A similar approach is to use different mappings in different blocks of the host signal.

The watermark embedding process 106 converts the message to a watermark information signal. It then combines this signal with the input signal and possibly another signal (e.g., an orientation pattern) to create a watermarked signal 108. The process of combining the watermark with the input signal may be a linear or non-linear function. Examples of watermarking functions include: $S^* = S + gX$; $S^* = S(1 + gX)$; and $S^* = S\, e^{gX}$; where S* is the watermarked signal vector, S is the input signal vector, and g is a function controlling watermark intensity. The watermark may be applied by modulating signal samples S in the spatial, temporal or some other transform domain.

To encode a message, the watermark encoder analyzes and selectively adjusts the host signal to give it attributes that correspond to the desired message symbol or symbols to be encoded. There are many signal attributes that may encode a message symbol, such as a positive or negative polarity of signal samples or a set of samples, a given parity (odd or even), a given difference value or polarity of the difference between signal samples (e.g., a difference between selected spatial intensity values or transform coefficients), a given distance value between watermarks, a given phase or phase offset between different watermark components, a modulation of the phase of the host signal, a modulation of frequency coefficients of the host signal, a given frequency pattern, a given quantizer (e.g., in Quantization Index Modulation) etc.

Some processes for combining the watermark with the input signal are termed non-linear, such as processes that employ dither modulation, modify least significant bits, or apply quantization index modulation. One type of non-linear modulation is where the embedder sets signal values so that they have some desired value or characteristic corresponding to a message symbol. For example, the embedder may designate that a portion of the host signal is to encode a given bit value. It then evaluates a signal value or set of values in that portion to determine whether they have the attribute corresponding to the message bit to be encoded. Some examples of attributes include a positive or negative polarity, a value that is odd or even, a checksum, etc. For example, a bit value may be encoded as a one or zero by quantizing the value of a selected sample to be even or odd. As another example, the embedder might compute a checksum or parity of an N bit pixel value or transform coefficient and then set the least significant bit to the value of the checksum or parity. Of course, if the signal already corresponds to the desired message bit value, it need not be altered. The same approach can be extended to a set of signal samples where some attribute of the set is adjusted as necessary to encode a desired message symbol. These techniques can be applied to signal samples in a transform domain (e.g., transform coefficients) or samples in the temporal or spatial domains.

Quantization index modulation techniques employ a set of quantizers. In these techniques, the message to be transmitted is used as an index for quantizer selection. In the decoding process, a distance metric is evaluated for all quantizers and the index with the smallest distance identifies the message value.

The watermark detector 110 operates on a digitized signal suspected of containing a watermark. As depicted generally in Fig. 1, the suspect signal may undergo various transformations 112, such as conversion to and from an analog domain, cropping, copying, editing, compression/decompression, transmission etc. Using parameters 114 from the embedder (e.g., orientation pattern, control bits, key(s)), it performs a series of correlation or other operations on the captured image to detect the presence of a watermark. If it finds a watermark, it determines its orientation within the suspect signal.

Using the orientation, if necessary, the reader 116 extracts the message. Some implementations do not perform correlation, but instead, use some other detection process or

proceed directly to extract the watermark signal. For instance in some applications, a reader may be invoked one or more times at various temporal or spatial locations in an attempt to read the watermark, without a separate pre-processing stage to detect the watermark's orientation.

Some implementations require the original, un-watermarked signal to decode a watermark message, while others do not. In those approaches where the original signal is not necessary, the original un-watermarked signal can still be used to improve the accuracy of message recovery. For example, the original signal can be removed, leaving a residual signal from which the watermark message is recovered. If the decoder does not have the original signal, it can still attempt to remove portions of it (e.g., by filtering) that are expected not to contain the watermark signal.

Watermark decoder implementations use known relationships between a watermark signal and a message symbol to extract estimates of message symbol values from a signal suspected of containing a watermark. The decoder has knowledge of the properties of message symbols and how and where they are encoded into the host signal to encode a message. For example, it knows how message bit values of one and a zero are encoded and it knows where these message bits are originally encoded. Based on this information, it can look for the message properties in the watermarked signal. For example, it can test the watermarked signal to see if it has attributes of each message symbol (e.g., a one or zero) at a particular location and generate a probability measure as an indicator of the likelihood that a message symbol has been encoded. Knowing the approximate location of the watermark in the watermarked signal, the reader implementation may compare known message properties with the properties of the watermarked signal to estimate message values, even if the original signal is unavailable. Distortions to the watermarked signal and the host signal itself make the watermark difficult to recover, but accurate recovery of the message can be enhanced using a variety of techniques, such as error correction coding, watermark signal prediction, redundant message encoding, etc.

One way to recover a message value from a watermarked signal is to perform correlation between the known message property of each message symbol and the watermarked signal. If the amount of correlation exceeds a threshold, for example, then the

watermarked signal may be assumed to contain the message symbol. The same process can be repeated for different symbols at various locations to extract a message. A symbol (e.g., a binary value of one or zero) or set of symbols may be encoded redundantly to enhance message recovery.

In some cases, it is useful to filter the watermarked signal to remove aspects of the signal that are unlikely to be helpful in recovering the message and/or are likely to interfere with the watermark message. For example, the decoder can filter out portions of the original signal and another watermark signal or signals. In addition, when the original signal is unavailable, the reader can estimate or predict the original signal based on properties of the watermarked signal. The original or predicted version of the original signal can then be used to recover an estimate of the watermark message. One way to use the predicted version to recover the watermark is to remove the predicted version before reading the desired watermark. Similarly, the decoder can predict and remove un-wanted watermarks or watermark components before reading the desired watermark in a signal having two or more watermarks.

2.1 Image Watermark Embedder

Fig. 2 is a block diagram illustrating an implementation of an exemplary embedder in more detail. The embedding process begins with the message 200. As noted above, the message is binary number suitable for conversion to a watermark signal. For additional security, the message, its carrier, and the mapping of the watermark to the host signal may be encrypted with an encryption key 202. In addition to the information conveyed in the message, the embedder may also add control bit values ("signature bits") to the message to assist in verifying the accuracy of a read operation. These control bits, along with the bits representing the message, are input to an error correction coding process 204 designed to increase the likelihood that the message can be recovered accurately in the reader.

There are several alternative error correction coding schemes that may be employed. Some examples include BCH, convolution, Reed Solomon and turbo codes. These forms of error correction coding are sometimes used in communication applications where data is encoded in a carrier signal that transfers the encoded data from one place to another. In the

digital watermarking application discussed here, the raw bit data is encoded in a fundamental carrier signal.

In addition to the error correction coding schemes mentioned above, the embedder and reader may also use a Cyclic Redundancy Check (CRC) to facilitate detection of errors in the decoded message data.

The error correction coding function 204 produces a string of bits, termed raw bits 206, that are embedded into a watermark information signal. Using a carrier signal 208 and an assignment map 210, the illustrated embedder encodes the raw bits in a watermark information signal 212, 214. In some applications, the embedder may encode a different message in different locations of the signal. The carrier signal may be a noise image. For each raw bit, the assignment map specifies the corresponding image sample or samples that will be modified to encode that bit.

The embedder depicted in Fig. 2 operates on blocks of image data (referred to as 'tiles') and replicates a watermark in each of these blocks. As such, the carrier signal and assignment map both correspond to an image block of a pre-determined size, namely, the size of the tile. To encode each bit, the embedder applies the assignment map to determine the corresponding image samples in the block to be modified to encode that bit. Using the map, it finds the corresponding image samples in the carrier signal. For each bit, the embedder computes the value of image samples in the watermark information signal as a function of the raw bit value and the value(s) of the corresponding samples in the carrier signal.

To illustrate the embedding process further, it is helpful to consider an example. First, consider the following background. Digital watermarking processes are sometimes described in terms of the transform domain in which the watermark signal is defined. The watermark may be defined in the spatial or temporal domain, or some other transform domain such as a wavelet transform, Discrete Cosine Transform (DCT), Discrete Fourier Transform (DFT), Hadamard transform, Hartley transform, Karhunen-Loeve transform (KLT) domain, etc.

Consider an example where the watermark is defined in a transform domain (e.g., a frequency domain such as DCT, wavelet or DFT). The embedder segments the image in the spatial domain into rectangular tiles and transforms the image samples in each tile into the

transform domain. For example in the DCT domain, the embedder segments the image into N by N blocks and transforms each block into an N by N block of DCT coefficients. In this example, the assignment map specifies the corresponding sample location or locations in the frequency domain of the tile that correspond to a bit position in the raw bits. In the frequency domain, the carrier signal looks like a noise pattern. Each image sample in the frequency domain of the carrier signal is used together with a selected raw bit value to compute the value of the image sample at the location in the watermark information signal.

Now consider an example where the watermark is defined in the spatial domain. The embedder segments the image in the spatial domain into rectangular tiles of image samples (i.e. pixels). In this example, the assignment map specifies the corresponding sample location or locations in the tile that correspond to each bit position in the raw bits. In the spatial domain, the carrier signal looks like a noise pattern extending throughout the tile. Each image sample in the spatial domain of the carrier signal is used together with a selected raw bit value to compute the value of the image sample at the same location in the watermark information signal.

With this background, the embedder proceeds to encode each raw bit in the selected transform domain as follows. It uses the assignment map to look up the position of the corresponding image sample (or samples) in the carrier signal. The image sample value at that position in the carrier controls the value of the corresponding position in the watermark information signal. In particular, the carrier sample value indicates whether to invert the corresponding watermark sample value. The raw bit value is either a one or zero. Disregarding for a moment the impact of the carrier signal, the embedder adjusts the corresponding watermark sample upward to represent a one, or downward to represent a zero. Now, if the carrier signal indicates that the corresponding sample should be inverted, the embedder adjusts the watermark sample downward to represent a one, and upward to represent a zero. In this manner, the embedder computes the value of the watermark samples for a raw bit using the assignment map to find the spatial location of those samples within the block.

From this example, a number of points can be made. First, the embedder may perform a similar approach in any other transform domain. Second, for each raw bit, the

corresponding watermark sample or samples are some function of the raw bit value and the carrier signal value. The specific mathematical relationship between the watermark sample, on one hand, and the raw bit value and carrier signal, on the other, may vary with the implementation. For example, the message may be convolved with the carrier, multiplied with the carrier, added to the carrier, or applied based on another non-linear function. Third, the carrier signal may remain constant for a particular application, or it may vary from one message to another. For example, a secret key may be used to generate the carrier signal. For each raw bit, the assignment map may define a pattern of watermark samples in the transform domain in which the watermark is defined. An assignment map that maps a raw bit to a sample location or set of locations (i.e. a map to locations in a frequency or spatial domain) is just one special case of an assignment map for a transform domain. Fourth, the assignment map may remain constant, or it may vary from one message to another. In addition, the carrier signal and map may vary depending on the nature of the underlying image. In sum, there many possible design choices within the implementation framework described above.

The embedder depicted in Fig. 2 combines another watermark component, shown as the detection watermark 216, with the watermark information signal to compute the final watermark signal. The detection watermark is specifically chosen to assist in identifying the watermark and computing its orientation in a detection operation.

Fig. 3 is a spatial frequency plot illustrating one quadrant of a detection watermark. The points in the plot represent impulse functions indicating signal content of the detection watermark signal. The pattern of impulse functions for the illustrated quadrant is replicated in all four quadrants. There are a number of properties of the detection pattern that impact its effectiveness for a particular application. The selection of these properties is highly dependent on the application. One property is the extent to which the pattern is symmetric about one or more axes. For example, if the detection pattern is symmetrical about the horizontal and vertical axes, it is referred to as being quad symmetric. If it is further symmetrical about diagonal axes at an angle of 45 degrees, it is referred to as being octally symmetric (repeated in a symmetric pattern 8 times about the origin). Such symmetry aids in identifying the watermark in an image, and aids in extracting the rotation angle. However, in

the case of an octally symmetric pattern, the detector includes an additional step of testing which of the four quadrants the orientation angle falls into.

Another criterion is the position of the impulse functions and the frequency range that they reside in. Preferably, the impulse functions fall in a mid frequency range. If they are located in a low frequency range, they may be noticeable in the watermarked image. If they are located in the high frequency range, they are more difficult to recover. Also, they should be selected so that scaling, rotation, and other manipulations of the watermarked signal do not push the impulse functions outside the range of the detector. Finally, the impulse functions should preferably not fall on the vertical or horizontal axes, and each impulse function should have a unique horizontal and vertical location. While the example depicted in Fig. 3 shows that some of the impulse functions fall on the same horizontal axis, it is trivial to alter the position of the impulse functions such that each has a unique vertical or horizontal coordinate.

Returning to Fig. 2, the embedder makes a perceptual analysis 218 of the input image 220 to identify portions of the image that can withstand more watermark signal content without substantially impacting image fidelity. Generally, the perceptual analysis employs a HVS model to identify signal frequency bands and/or spatial areas to increase or decrease watermark signal intensity to make the watermark imperceptible to an ordinary observer. One type of model is to increase watermark intensity in frequency bands and spatial areas where there is more image activity. In these areas, the sample values are changing more than other areas and have more signal strength. The output of the perceptual analysis is a perceptual mask 222. The mask may be implemented as an array of functions, which selectively increase the signal strength of the watermark signal based on a HVS model analysis of the input image. The mask may selectively increase or decrease the signal strength of the watermark signal in areas of greater signal activity.

The embedder combines (224) the watermark information, the detection signal and the perceptual mask to yield the watermark signal 226. Finally, it combines (228) the input image 220 and the watermark signal 226 to create the watermarked image 230. In the frequency domain watermark example above, the embedder combines the transform domain coefficients in the watermark signal to the corresponding coefficients in the input image to

create a frequency domain representation of the watermarked image. It then transforms the image into the spatial domain. As an alternative, the embedder may be designed to convert the watermark into the spatial domain, and then add it to the image.

In the spatial watermark example above, the embedder combines the image samples in the watermark signal to the corresponding samples in the input image to create the watermarked image 230.

The embedder may employ an invertible or non-invertible, and linear or non-linear function to combine the watermark signal and the input image (e.g., linear functions such as $S^* = S + gX$; or $S^* = S(1 + gX)$, convolution, quantization index modulation). The net effect is that some image samples in the input image are adjusted upward, while others are adjusted downward. The extent of the adjustment is greater in areas or subbands of the image having greater signal activity.

2.2. Overview of a Detector and Reader

Fig. 4 is a flow diagram illustrating an overview of a watermark detection process. This process analyzes image data 400 to search for an orientation pattern of a watermark in an image suspected of containing the watermark (the target image). First, the detector transforms the image data to another domain 402, namely the spatial frequency domain, and then performs a series of correlation or other detection operations 404. The correlation operations match the orientation pattern with the target image data to detect the presence of the watermark and its orientation parameters 406 (e.g., translation, scale, rotation, and differential scale relative to its original orientation). Finally, it re-orients the image data based on one or more of the orientation parameters 408.

If the orientation of the watermark is recovered, the reader extracts the watermark information signal from the image data (optionally by first re-orienting the data based on the orientation parameters). Fig. 5 is flow diagram illustrating a process of extracting a message from re-oriented image data 500. The reader scans the image samples (e.g., pixels or transform domain coefficients) of the re-oriented image (502), and uses known attributes of the watermark signal to estimate watermark signal values 504. Recall that in one example implementation described above, the embedder adjusted sample values (e.g., frequency

coefficients, color values, etc.) up or down to embed a watermark information signal. The reader uses this attribute of the watermark information signal to estimate its value from the target image. Prior to making these estimates, the reader may filter the image to remove portions of the image signal that may interfere with the estimating process. For example, if the watermark signal is expected to reside in low or medium frequency bands, then high frequencies may be filtered out.

In addition, the reader may predict the value of the original un-watermarked image to enhance message recovery. One form of prediction uses temporal or spatial neighbors to estimate a sample value in the original image. In the frequency domain, frequency coefficients of the original signal can be predicted from neighboring frequency coefficients in the same frequency subband. In video applications for example, a frequency coefficient in a frame can be predicted from spatially neighboring coefficients within the same frame, or temporally neighboring coefficients in adjacent frames or fields. In the spatial domain, intensity values of a pixel can be estimated from intensity values of neighboring pixels. Having predicted the value of a signal in the original, un-watermarked image, the reader then estimates the watermark signal by calculating an inverse of the watermarking function used to combine the watermark signal with the original signal.

For such watermark signal estimates, the reader uses the assignment map to find the corresponding raw bit position and image sample in the carrier signal (506). The value of the raw bit is a function of the watermark signal estimate, and the carrier signal at the corresponding location in the carrier. To estimate the raw bit value, the reader solves for its value based on the carrier signal and the watermark signal estimate. As reflected generally in Fig. 5 (508), the result of this computation represents only one estimate to be analyzed along with other estimates impacting the value of the corresponding raw bit. Some estimates may indicate that the raw bit is likely to be a one, while others may indicate that it is a zero. After the reader completes its scan, it compiles the estimates for each bit position in the raw bit string, and makes a determination of the value of each bit at that position (510). Finally, it performs the inverse of the error correction coding scheme to construct the message (512). In some implementations, probablistic models may be employed to determine the likelihood that a particular pattern of raw bits is just a random occurrence rather than a watermark.

2.2.1 Example Illustrating Detector Process

Fig. 6 is a diagram depicting an example of a watermark detection process. The detector segments the target image into blocks (e.g., 600, 602) and then performs a 2-dimensional fast fourier transform (2D FFT) on several blocks. This process yields 2D transforms of the magnitudes of the image contents of the blocks in the spatial frequency domain as depicted in the plot 604 shown in Fig. 6.

Next, the detector process performs a log polar remapping of each transformed block. The detector may add some of the blocks together to increase the watermark signal to noise ratio. The type of remapping in this implementation is referred to as a Fourier Mellin transform. The Fourier Mellin transform is a geometric transform that warps the image data from a frequency domain to a log polar coordinate system. As depicted in the plot 606 shown in Fig. 6, this transform sweeps through the transformed image data along a line at angle θ, mapping the data to a log polar coordinate system shown in the next plot 608. The log polar coordinate system has a rotation axis, representing the angle θ, and a scale axis. Inspecting the transformed data at this stage, one can see the orientation pattern of the watermark begin to be distinguishable from the noise component (i.e., the image signal).

Next, the detector performs a correlation 610 between the transformed image block and the transformed orientation pattern 612. At a high level, the correlation process slides the orientation pattern over the transformed image (in a selected transform domain, such as a spatial frequency domain) and measures the correlation at an array of discrete positions. Each such position has a corresponding scale and rotation parameter associated with it. Ideally, there is a position that clearly has the highest correlation relative to all of the others. In practice, there may be several candidates with a promising measure of correlation. As explained further below, these candidates may be subjected to one or more additional correlation stages to select the one that provides the best match.

There are a variety of ways to implement the correlation process. Any number of generalized matching filters may be implemented for this purpose. One such filter performs an FFT on the target and the orientation pattern, and multiplies the resulting arrays together to yield a multiplied FFT. Finally, it performs an inverse FFT on the multiplied FFT to

return the data into its original log-polar domain. The position or positions within this resulting array with the highest magnitude represent the candidates with the highest correlation.

When there are several viable candidates, the detector can select a set of the top candidates and apply an additional correlation stage. Each candidate has a corresponding rotation and scale parameter. The correlation stage rotates and scales the FFT of the orientation pattern and performs a matching operation with the rotated and scaled pattern on the FFT of the target image. The matching operation multiplies the values of the transformed pattern with sample values at corresponding positions in the target image and accumulates the result to yield a measure of the correlation. The detector repeats this process for each of the candidates and picks the one with the highest measure of correlation. As shown in Fig. 6, the rotation and scale parameters (614) of the selected candidate are then used to find additional parameters that describe the orientation of the watermark in the target image.

The detector applies the scale and rotation to the target data block 616 and then performs another correlation process between the orientation pattern 618 and the scaled and rotated data block 616. The correlation process 620 is a generalized matching filter operation. It provides a measure of correlation for an array of positions that each has an associated translation parameter (e.g., an x, y position). Again, the detector may repeat the process of identifying promising candidates (i.e. those that reflect better correlation relative to others) and using those in an additional search for a parameter or set of orientation parameters that provide a better measure of correlation.

At this point, the detector has recovered the following orientation parameters: rotation, scale and translation. For many applications, these parameters may be sufficient to enable accurate reading of the watermark. In the read operation, the reader applies the orientation parameters to re-orient the target image and then proceeds to extract the watermark signal.

In some applications, the watermarked image may be stretched more in one spatial dimension than another. This type of distortion is sometimes referred to as differential scale or shear. Consider that the original image blocks are square. As a result of differential scale,

each square may be warped into a parallelogram with unequal sides. Differential scale parameters define the nature and extent of this stretching.

There are several alternative ways to recover the differential scale parameters. One general class of techniques is to use the known parameters (e.g., the computed scale, rotation, and translation) as a starting point to find the differential scale parameters. Assuming the known parameters to be valid, this approach warps either the orientation pattern or the target image with selected amounts of differential scale and picks the differential scale parameters that yield the best correlation.

Another approach to determination of differential scale is set forth in application 09/452,022 (filed November 30, 1999, and entitled Method and System for Determining Image Transformation, attorney docket 60057).

2.2.2 Example Illustrating Reader Process

Fig. 7 is a diagram illustrating a re-oriented image 700 superimposed onto the original watermarked image 702. The difference in orientation and scale shows how the image was transformed and edited after the embedding process. The original watermarked image is sub-divided into tiles (e.g., pixel blocks 704, 706, etc.). When superimposed on the coordinate system of the original image 702 shown in Fig. 7, the target image blocks typically do not match the orientation of the original blocks.

The reader scans samples of the re-oriented image data, estimating the watermark information signal. It estimates the watermark information signal, in part, by predicting original sample values of the un-watermarked image. The reader then uses an inverted form of the watermarking function to estimate the watermark information signal from the watermarked signal and the predicted signal. This inverted watermarking function expresses the estimate of the watermark signal as a function of the predicted signal and the watermarked signal. Having an estimate of the watermark signal, it then uses the known relationship among the carrier signal, the watermark signal, and the raw bit to compute an estimate of the raw bit. Recall that samples in the watermark information signal are a function of the carrier signal and the raw bit value. Thus, the reader may invert this function to solve for an estimate of the raw bit value.

Recall that the embedder implementation discussed in connection with Fig. 2 redundantly encodes the watermark information signal in blocks of the input signal. Each raw bit may map to several samples within a block. In addition, the embedder repeats a mapping process for each of the blocks. As such, the reader generates several estimates of the raw bit value as it scans the watermarked image.

The information encoded in the raw bit string can be used to increase the accuracy of read operations. For instance, some of the raw bits act as signature bits that perform a validity checking function. Unlike unknown message bits, the reader knows the expected values of these signature bits. The reader can assess the validity of a read operation based on the extent to which the extracted signature bit values match the expected signature bit values. The estimates for a given raw bit value can then be given a higher weight depending on whether they are derived from a tile with a greater measure of validity.

3.0 Embedder Implementation:

The following sections describe an implementation of the digital image watermark embedder depicted in Fig. 8. The embedder inserts two watermark components into the host image: a message component and a detection component (called the orientation pattern). The message component is defined in a spatial domain or other transform domain, while the orientation pattern is defined in a frequency domain. As explained later, the message component serves a dual function of conveying a message and helping to identify the watermark location in the image.

The embedder inserts the watermark message and orientation pattern in blocks of a selected color plane or planes (e.g., luminance or chrominance plane) of the host image. The message payload varies from one application to another, and can range from a single bit to the number of image samples in the domain in which it is embedded. The blocks may be blocks of samples in a spatial domain or some other transform domain.

3.1 Encoding the Message

The embedder converts binary message bits into a series of binary raw bits that it hides in the host image. As part of this process, a message encoder 800 appends certain

known bits to the message bits 802. It performs an error detection process (e.g., parity, Cyclic Redundancy Check (CRC), etc.) to generate error detection bits and adds the error detection bits to the message. An error correction coding operation then generates raw bits from the combined known and message bit string.

For the error correction operation, the embedder may employ any of a variety of error correction codes such as Reed Solomon, BCH, convolution or turbo codes. The encoder may perform an M-ary modulation process on the message bits that maps groups of message bits to a message signal based on an M-ary symbol alphabet.

In one application of the embedder, the component of the message representing the known bits is encoded more redundantly than the other message bits. This is an example of a shorter message component having greater signal strength than a longer, weaker message component. The embedder gives priority to the known bits in this scheme because the reader uses them to verify that it has found the watermark in a potentially corrupted image, rather than a signal masquerading as the watermark.

3.2 Spread Spectrum Modulation

The embedder uses spread spectrum modulation as part of the process of creating a watermark signal from the raw bits. A spread spectrum modulator 804 spreads each raw bit into a number of "chips." The embedder generates a pseudo random number that acts as the carrier signal of the message. To spread each raw bit, the modulator performs an exclusive OR (XOR) operation between the raw bit and each bit of a pseudo random binary number of a pre-determined length. The length of the pseudo random number depends, in part, on the size of the message and the image. Preferably, the pseudo random number should contain roughly the same number of zeros and ones, so that the net effect of the raw bit on the host image block is zero. If a bit value in the pseudo random number is a one, the value of the raw bit is inverted. Conversely, if the bit value is a zero, then the value of the raw bit remains the same.

The length of the pseudorandom number may vary from one message bit or symbol to another. By varying the length of the number, some message bits can be spread more than others.

3.3 Scattering the Watermark Message

The embedder scatters each of the chips corresponding to a raw bit throughout an image block. An assignment map 806 assigns locations in the block to the chips of each raw bit. Each raw bit is spread over several chips. As noted above, an image block may represent a block of transform domain coefficients or samples in a spatial domain. The assignment map may be used to encode some message bits or symbols (e.g., groups of bits) more redundantly than others by mapping selected bits to more locations in the host signal than other message bits. In addition, it may be used to map different messages, or different components of the same message, to different locations in the host signal.

Fig. 9 depicts an example of the assignment map. Each of the blocks in Fig. 9 correspond to an image block and depict a pattern of chips corresponding to a single raw bit. Fig. 9 depicts a total of 32 example blocks. The pattern within a block is represented as black dots on a white background. Each of the patterns is mutually exclusive such that each raw bit maps to a pattern of unique locations relative to the patterns of every other raw bit. Though not a requirement, the combined patterns, when overlapped, cover every location within the image block.

3.4 Gain Control and Perceptual Analysis

To insert the information carried in a chip to the host image, the embedder alters the corresponding sample value in the host image. In particular, for a chip having a value of one, it adds to the corresponding sample value, and for a chip having a value of zero, it subtracts from the corresponding sample value. A gain controller in the embedder adjusts the extent to which each chip adds or subtracts from the corresponding sample value.

The gain controller takes into account the orientation pattern when determining the gain. It applies a different gain to the orientation pattern than to the message component of the watermark. After applying the gain, the embedder combines the orientation pattern and message components together to form the composite watermark signal, and combines the composite watermark with the image block. One way to combine these signal components is to add them, but other linear or non-linear functions may be used as well.

The orientation pattern is comprised of a pattern of quad symmetric impulse functions in the spatial frequency domain. In the spatial domain, these impulse functions look like cosine waves. An example of the orientation pattern is depicted in Figs. 10 and 11. Fig. 10 shows the impulse functions as points in the spatial frequency domain, while Fig. 11 shows the orientation pattern in the spatial domain. Before adding the orientation pattern component to the message component, the embedder may transform the watermark components to a common domain. For example, if the message component is in a spatial domain and the orientation component is in a frequency domain, the embedder transforms the orientation component to a common spatial domain before combining them together.

Fig. 8 depicts the gain controller used in the embedder. Note that the gain controller operates on the blocks of image samples 808, the message watermark signal, and a global gain input 810, which may be specified by the user. A perceptual analyzer component 812 of the gain controller performs a perceptual analysis on the block to identify samples that can tolerate a stronger watermark signal without substantially impacting visibility. In places where the naked eye is less likely to notice the watermark, the perceptual analyzer increases the strength of the watermark. Conversely, it decreases the watermark strength where the eye is more likely to notice the watermark.

The perceptual analyzer shown in Fig. 8 performs a series of filtering operations on the image block to compute an array of gain values. There are a variety of filters suitable for this task. These filters include an edge detector filter that identifies edges of objects in the image, a non-linear filter to map gain values into a desired range, and averaging or median filters to smooth the gain values. Each of these filters may be implemented as a series of one-dimensional filters (one operating on rows and the other on columns) or two-dimensional filters. The size of the filters (i.e. the number of samples processed to compute a value for a given location) may vary (e.g., 3 by 3, 5 by 5, etc.). The shape of the filters may vary as well (e.g., square, cross-shaped, etc.). The perceptual analyzer process produces a detailed gain multiplier. The multiplier is a vector with elements corresponding to samples in a block.

Another component 818 of the gain controller computes an asymmetric gain based on the output of the image sample values and message watermark signal.

This component analyzes the samples of the block to determine whether they are consistent with the message signal. The embedder reduces the gain for samples whose values relative to neighboring values are consistent with the message signal.

The embedder applies the asymmetric gain to increase the chances of an accurate read in the watermark reader. To understand the effect of the asymmetric gain, it is helpful to explain the operation of the reader. The reader extracts the watermark message signal from the watermarked signal using a predicted version of the original signal. It estimates the watermark message signal value based on values of the predicted signal and the watermarked signal at locations of the watermarked signal suspected of containing a watermark signal. There are several ways to predict the original signal. One way is to compute a local average of samples around the sample of interest. The average may be computed by taking the average of vertically adjacent samples, horizontally adjacent samples, an average of samples in a cross-shaped filter (both vertical and horizontal neighbors, an average of samples in a square-shaped filter, etc. The estimate may be computed one time based on a single predicted value from one of these averaging computations. Alternatively, several estimates may be computed based on two or more of these averaging computations (e.g., one estimate for vertically adjacent samples and another for horizontally adjacent samples). In the latter case, the reader may keep estimates if they satisfy a similarity metric. In other words, the estimates are deemed valid if they within a predetermined value or have the same polarity.

Knowing this behavior of the reader, the embedder computes the asymmetric gain as follows. For samples that have values relative to their neighbors that are consistent with the watermark signal, the embedder reduces the asymmetric gain. Conversely, for samples that are inconsistent with the watermark signal, the embedder increases the asymmetric gain. For example, if the chip value is a one, then the sample is consistent with the watermark signal if its value is greater than its neighbors. Alternatively, if the chip value is a zero, then the sample is consistent with the watermark signal if its value is less than its neighbors.

Another component 820 of the gain controller computes a differential gain, which represents an adjustment in the message vs. orientation pattern gains. As the global gain increases, the embedder emphasizes the message gain over the orientation pattern gain by adjusting the global gain by an adjustment factor. The inputs to this process 820 include the

global gain 810 and a message differential gain 822. When the global gain is below a lower threshold, the adjustment factor is one. When the global gain is above an upper threshold, the adjustment factor is set to an upper limit greater than one. For global gains falling within the two thresholds, the adjustment factor increases linearly between one and the upper limit. The message differential gain is the product of the adjustment factor and the global gain.

At this point, there are four sources of gain: the detailed gain, the global gain, the asymmetric gain, and the message dependent gain. The embedder applies the first two gain quantities to both the message and orientation watermark signals. It only applies the latter two to the message watermark signal. Fig. 8 depicts how the embedder applies the gain to the two watermark components. First, it multiplies the detailed gain with the global gain to compute the orientation pattern gain. It then multiplies the orientation pattern gain with the adjusted message differential gain and asymmetric gain to form the composite message gain.

Finally, the embedder forms the composite watermark signal. It multiplies the composite message gain with the message signal, and multiplies the orientation pattern gain with the orientation pattern signal. It then combines the result in a common transform domain to get the composite watermark. The embedder applies a watermarking function to combine the composite watermark to the block to create a watermarked image block. The message and orientation components of the watermark may be combined by mapping the message bits to samples of the orientation signal, and modulating the samples of the orientation signal to encode the message.

The embedder computes the watermark message signal by converting the output of the assignment map 806 to delta values, indicating the extent to which the watermark signal changes the host signal. As noted above, a chip value of one corresponds to an upward adjustment of the corresponding sample, while a chip value of zero corresponds to a downward adjustment. The embedder specifies the specific amount of adjustment by assigning a delta value to each of the watermark message samples (830).

4.0 Detector Implementation

Fig. 12 illustrates an overview of a watermark detector that detects the presence of a detection watermark in a host image and its orientation. Using the orientation pattern and the

known bits inserted in the watermark message, the detector determines whether a potentially corrupted image contains a watermark, and if so, its orientation in the image.

Recall that the composite watermark is replicated in blocks of the original image. After an embedder places the watermark in the original digital image, the watermarked image is likely to undergo several transformations, either from routine processing or from intentional tampering. Some of these transformations include: compression, decompression, color space conversion, digital to analog conversion, printing, scanning, analog to digital conversion, scaling, rotation, inversion, flipping differential scale, and lens distortion. In addition to these transformations, various noise sources can corrupt the watermark signal, such as fixed pattern noise, thermal noise, etc.

When building a detector implementation for a particular application, the developer may implement counter-measures to mitigate the impact of the types of transformations, distortions and noise expected for that application. Some applications may require more counter-measures than others. The detector described below is designed to recover a watermark from a watermarked image after the image has been printed, and scanned. The following sections describe the counter-measures to mitigate the impact of various forms of corruption. The developer can select from among these counter-measures when implementing a detector for a particular application.

For some applications, the detector will operate in a system that provides multiple image frames of a watermarked object. One typical example of such a system is a computer equipped with a digital camera. In such a configuration, the digital camera can capture a temporal sequence of images as the user or some device presents the watermarked image to the camera.

As shown in Fig. 12, the principal components of the detector are: 1) pre-processor 900; 2) rotation and scale estimator 902; 3) orientation parameter refiner 904; 4) translation estimator 906; 5) translation refiner 908; and reader 910.

The preprocessor 900 takes one or more frames of image data 912 and produces a set of image blocks 914 prepared for further analysis. The rotation-scale estimator 902 computes rotation-scale vectors 916 that estimate the orientation of the orientation signal in the image blocks. The parameter refiner 904 collects additional evidence of the orientation

signal and further refines the rotation scale vector candidates by estimating differential scale parameters. The result of this refining stage is a set of 4D vectors candidates 918 (rotation, scale, and two differential scale parameters). The translation estimator 906 uses the 4D vector candidates to re-orient image blocks with promising evidence of the orientation signal. It then finds estimates of translation parameters 920. The translation refiner 908 invokes the reader 910 to assess the merits of an orientation vector. When invoked by the detector, the reader uses the orientation vector to approximate the original orientation of the host image and then extracts values for the known bits in the watermark message. The detector uses this information to assess the merits of and refine orientation vector candidates.

By comparing the extracted values of the known bits with the expected values, the reader provides a figure of merit for an orientation vector candidate. The translation refiner then picks a 6D vector, including rotation, scale, differential scale and translation, that appears likely produce a valid read of the watermark message 922. The following sections describe implementations of these components in more detail.

4.1 Detector Pre-processing

Fig. 13 is a flow diagram illustrating preprocessing operations in the detector shown in Fig. 12. The detector performs a series of pre-processing operations on the native image 930 to prepare the image data for further analysis. It begins by filling memory with one or more frames of native image data (932), and selecting sets of pixel blocks 934 from the native image data for further analysis (936). While the detector can detect a watermark using a single image frame, it also has support for detecting the watermark using additional image frames. As explained below, the use of multiple frames has the potential for increasing the chances of an accurate detection and read.

In applications where a camera captures an input image of a watermarked object, the detector may be optimized to address problems resulting from movement of the object. Typical PC cameras, for example, are capable of capturing images at a rate of at least 10 frames a second. A frustrated user might attempt to move the object in an attempt to improve detection. Rather than improving the chances of detection, the movement of the object changes the orientation of the watermark from one frame to the next, potentially making the watermark more difficult to detect. One way to address this problem is to buffer one or more

frames, and then screen the frame or frames to determine if they are likely to contain a valid watermark signal. If such screening indicates that a frame is not likely to contain a valid signal, the detector can discard it and proceed to the next frame in the buffer, or buffer a new frame. Another enhancement is to isolate portions of a frame that are most likely to have a valid watermark signal, and then perform more detailed detection of the isolated portions.

After loading the image into the memory, the detector selects image blocks 934 for further analysis. It is not necessary to load or examine each block in a frame because it is possible to extract the watermark using only a portion of an image. The detector looks at only a subset of the samples in an image, and preferably analyzes samples that are more likely to have a recoverable watermark signal.

The detector identifies portions of the image that are likely to have the highest watermark signal to noise ratio. It then attempts to detect the watermark signal in the identified portions. In the context of watermark detection, the host image is considered to be a source of noise along with conventional noise sources. While it is typically not practical to compute the signal to noise ratio, the detector can evaluate attributes of the signal that are likely to evince a promising watermark signal to noise ratio. These properties include the signal activity (as measured by sample variance, for example), and a measure of the edges (abrupt changes in image sample values) in an image block. Preferably, the signal activity of a candidate block should fall within an acceptable range, and the block should not have a high concentration of strong edges. One way to quantify the edges in the block is to use an edge detection filter (e.g., a LaPlacian, Sobel, etc.).

In one implementation, the detector divides the input image into blocks, and analyzes each block based on pre-determined metrics. It then ranks the blocks according to these metrics. The detector then operates on the blocks in the order of the ranking. The metrics include sample variance in a candidate block and a measure of the edges in the block. The detector combines these metrics for each candidate block to compute a rank representing the probability that it contains a recoverable watermark signal.

In another implementation, the detector selects a pattern of blocks and evaluates each one to try to make the most accurate read from the available data. In either implementation, the block pattern and size may vary. This particular implementation selects a pattern of

overlapping blocks (e.g., a row of horizontally aligned, overlapping blocks). One optimization of this approach is to adaptively select a block pattern that increases the signal to noise ratio of the watermark signal. While shown as one of the initial operations in the preparation, the selection of blocks can be postponed until later in the pre-processing stage.

Next, the detector performs a color space conversion on native image data to compute an array of image samples in a selected color space for each block (936). In the following description, the color space is luminance, but the watermark may be encoded in one or more different color spaces. The objective is to get a block of image samples with lowest noise practical for the application. While the implementation currently performs a row by row conversion of the native image data into 8 bit integer luminance values, it may be preferable to convert to floating-point values for some applications. One optimization is to select a luminance converter that is adapted for the sensor used to capture the digital input image. For example, one might experimentally derive the lowest noise luminance conversion for commercially available sensors, e.g., CCD cameras or scanners, CMOS cameras, etc. Then, the detector could be programmed to select either a default luminance converter, or one tuned to a specific type of sensor.

At one or more stages of the detector, it may be useful to perform operations to mitigate the impact of noise and distortion. In the pre-processing phase, for example, it may be useful to evaluate fixed pattern noise and mitigate its effect (938). The detector may look for fixed pattern noise in the native input data or the luminance data, and then mitigate it.

One way to mitigate certain types of noise is to combine data from different blocks in the same frame, or corresponding blocks in different frames 940. This process helps augment the watermark signal present in the blocks, while reducing the noise common to the blocks. For example, merely adding blocks together may mitigate the effects of common noise.

In addition to common noise, other forms of noise may appear in each of the blocks such as noise introduced in the printing or scanning processes. Depending on the nature of the application, it may be advantageous to perform common noise recognition and removal at this stage 942. The developer may select a filter or series of filters to target certain types of noise that appear during experimentation with images. Certain types of median filters may

be effective in mitigating the impact of spectral peaks (e.g., speckles) introduced in printing or scanning operations.

In addition to introducing noise, the printing and image capture processes may transform the color or orientation of the original, watermarked image. As described above, the embedder typically operates on a digital image in a particular color space and at a desired resolution. The watermark embedders normally operate on digital images represented in an RGB or CYMK color space at a desired resolution (e.g., 100 dpi or 300 dpi, the resolution at which the image is printed). The images are then printed on paper with a screen printing process that uses the CYMK subtractive color space at a line per inch (LPI) ranging from 65-200. 133 lines/in is typical for quality magazines and 73 lines/in is typical for newspapers. In order to produce a quality image and avoid pixelization, the rule of thumb is to use digital images with a resolution that is at least twice the press resolution. This is due to the half tone printing for color production. Also, different presses use screens with different patterns and line orientations and have different precision for color registration.

One way to counteract the transforms introduced through the printing process is to develop a model that characterizes these transforms and optimize watermark embedding and detecting based on this characterization. Such a model may be developed by passing watermarked and unwatermarked images through the printing process and observing the changes that occur to these images. The resulting model characterizes the changes introduced due to the printing process. The model may represent a transfer function that approximates the transforms due to the printing process. The detector then implements a pre-processing stage that reverses or at least mitigates the effect of the printing process on watermarked images. The detector may implement a pre-processing stage that performs the inverse of the transfer function for the printing process.

A related challenge is the variety in paper attributes used in different printing processes. Papers of various qualities, thickness and stiffness, absorb ink in various ways. Some papers absorb ink evenly, while others absorb ink at rates that vary with the changes in the paper's texture and thickness. These variations may degrade the embedded watermark signal when a digitally watermarked image is printed. The watermark process can counteract

these effects by classifying and characterizing paper so that the embedder and reader can compensate for this printing-related degradation.

Variations in image capture processes also pose a challenge. In some applications, it is necessary to address problems introduced due to interlaced image data. Some video camera produce interlaced fields representing the odd or even scan lines of a frame. Problems arise when the interlaced image data consists of fields from two consecutive frames. To construct an entire frame, the preprocessor may combine the fields from consecutive frames while dealing with the distortion due to motion that occurs from one frame to the next. For example, it may be necessary to shift one field before interleaving it with another field to counteract inter-frame motion. A de-blurring function may be used to mitigate the blurring effect due to the motion between frames.

Another problem associated with cameras in some applications is blurring due to the lack of focus. The preprocessor can mitigate this effect by estimating parameters of a blurring function and applying a de-blurring function to the input image.

Yet another problem associated with cameras is that they tend to have color sensors that utilize different color pattern implementations. As such, a sensor may produce colors slightly different than those represented in the object being captured. Most CCD and CMOS cameras use an array of sensors to produce colored images. The sensors in the array are arranged in clusters of sensitive to three primary colors red, green, and blue according to a specific pattern. Sensors designated for a particular color are dyed with that color to increase their sensitivity to the designated color. Many camera manufacturers use a Bayer color pattern GR/BG. While this pattern produces good image quality, it causes color mis-registration that degrades the watermark signal. Moreover, the color space converter, which maps the signal from the sensors to another color space such as YUV or RGB, may vary from one manufacturer to another. One way to counteract the mis-registration of the camera's color pattern is to account for the distortion due to the pattern in a color transformation process, implemented either within the camera itself, or as a pre-processing function in the detector.

Another challenge in counteracting the effects of the image capture process is dealing with the different types of distortion introduced from various image capture devices. For

example, cameras have different sensitivities to light. In addition, their lenses have different spherical distortion, and noise characteristics. Some scanners have poor color reproduction or introduce distortion in the image aspect ratio. Some scanners introduce aliasing and employ interpolation to increase resolution. The detector can counteract these effects in the pre-processor by using an appropriate inverse transfer function. An off-line process first characterizes the distortion of several different image capture devices (e.g., by passing test images through the scanner and deriving a transfer function modeling the scanner distortion). Some detectors may be equipped with a library of such inverse transfer functions from which they select one that corresponds to the particular image capture device

Yet another challenge in applications where the image is printed on paper and later scanned is that the paper deteriorates over time and degrades the watermark. Also, varying lighting conditions make the watermark difficult to detect. Thus, the watermark may be selected so as to be more impervious to expected deterioration, and recoverable over a wider range of lighting conditions.

At the close of the pre-processing stage, the detector has selected a set of blocks for further processing. It then proceeds to gather evidence of the orientation signal in these blocks, and estimate the orientation parameters of promising orientation signal candidates. Since the image may have suffered various forms of corruption, the detector may identify several parts of the image that appear to have attributes similar to the orientation signal. As such, the detector may have to resolve potentially conflicting and ambiguous evidence of the orientation signal. To address this challenge, the detector estimates orientation parameters, and then refines theses estimates to extract the orientation parameters that are more likely to evince a valid signal than other parameter candidates.

4.2 Estimating Initial Orientation Parameters

Fig. 14 is a flow diagram illustrating a process for estimating rotation-scale vectors. The detector loops over each image block (950), calculating rotation-scale vectors with the best detection values in each block. First, the detector filters the block in a manner that tends to amplify the orientation signal while suppressing noise, including noise from the host image itself (952). Implemented as a multi-axis LaPlacian filter, the filter highlights edges

(e.g., high frequency components of the image) and then suppresses them. The term, "multi-axis," means that the filter includes a series of stages that each operates on particular axis. First, the filter operates on the rows of luminance samples, then operates on the columns, and adds the results. The filter may be applied along other axes as well. Each pass of the filter produces values at discrete levels. The final result is an array of samples, each having one of five values: {-2, -1, 0, 1, 2}.

Next, the detector performs a windowing operation on the block data to prepare it for an FFT transform (954). This windowing operation provides signal continuity at the block edges. The detector then performs an FFT (956) on the block, and retains only the magnitude component (958).

In an alternative implementation, the detector may use the phase signal produced by the FFT to estimate the translation parameter of the orientation signal. For example, the detector could use the rotation and scale parameters extracted in the process described below, and then compute the phase that provided the highest measure of correlation with the orientation signal using the phase component of the FFT process.

After computing the FFT, the detector applies a Fourier magnitude filter (960) on the magnitude components. The filter in the implementation slides over each sample in the Fourier magnitude array and filters the sample's eight neighbors in a square neighborhood centered at the sample. The filter boosts values representing a sharp peak with a rapid fall-off, and suppresses the fall-off portion. It also performs a threshold operation to clip peaks to an upper threshold.

Next, the detector performs a log-polar re-sample (962) of the filtered Fourier magnitude array to produce a log-polar array 964. This type of operation is sometimes referred to as a Fourier Mellin transform. The detector, or some off-line pre-processor, performs a similar operation on the orientation signal to map it to the log-polar coordinate system. Using matching filters, the detector implementation searches for a orientation signal in a specified window of the log-polar coordinate system. For example, consider that the log-polar coordinate system is a two dimensional space with the scale being the vertical axis and the angle being the horizontal axis. The window ranges from 0 to 90 degrees on the horizontal axis and from approximately 50 to 2400 dpi on the vertical axis. Note that the

orientation pattern should be selected so that routine scaling does not push the orientation pattern out of this window. The orientation pattern can be designed to mitigate this problem, as noted above, and as explained in co-pending patent application no. 60/136,572, filed May 28, 1999, by Ammon Gustafson, entitled Watermarking System With Improved Technique for Detecting Scaling and Rotation, filed May 28, 1999.

The detector proceeds to correlate the orientation and the target signal in the log polar coordinate system. As shown in Fig. 14, the detector uses a generalized matched filter GMF (966). The GMF performs an FFT on the orientation and target signal, multiplies the resulting Fourier domain entities, and performs an inverse FFT. This process yields a rectangular array of values in log-polar coordinates, each representing a measure of correlation and having a corresponding rotation angle and scale vector. As an optimization, the detector may also perform the same correlation operations for distorted versions (968, 970, 972) of the orientation signal to see if any of the distorted orientation patterns results in a higher measure of correlation. For example, the detector may repeat the correlation operation with some pre-determined amount of horizontal and vertical differential distortion (970, 972). The result of this correlation process is an array of correlation values 974 specifying the amount of correlation that each corresponding rotation-scale vector provides.

The detector processes this array to find the top M peaks and their location in the log-polar space 976. To extract the location more accurately, the detector uses interpolation to provide the inter-sample location of each of the top peaks 978. The interpolator computes the 2D median of the samples around a peak and provides the location of the peak center to an accuracy of 0.1 sample.

The detector proceeds to rank the top rotation-scale vectors based on yet another correlation process 980. In particular, the detector performs a correlation between a Fourier magnitude representation for each rotation-scale vector candidate and a Fourier magnitude specification of the orientation signal 982. Each Fourier magnitude representation is scaled and rotated by an amount reflected by the corresponding rotation-scale vector. This correlation operation sums a point-wise multiplication of the orientation pattern impulse functions in the frequency domain with the Fourier magnitude values of the image at

corresponding frequencies to compute a measure of correlation for each peak 984. The detector then sorts correlation values for the peaks (986).

Finally, the detector computes a detection value for each peak (988). It computes the detection value by quantizing the correlation values. Specifically, it computes a ratio of the peak's correlation value and the correlation value of the next largest peak. Alternatively, the detector may compute the ratio of the peak's correlation value and a sum or average of the correlation values of the next n highest peaks, where n is some predetermined number. Then, the detector maps this ratio to a detection value based on a statistical analysis of unmarked images.

The statistical analysis plots a distribution of peak ratio values found in unmarked images. The ratio values are mapped to a detection value based on the probability that the value came from an unmarked image. For example, 90% of the ratio values in unmarked images fall below a first threshold T1, and thus, the detection value mapping for a ratio of T1 is set to 1. Similarly, 99% of the ratio values in unmarked images fall below T2, and therefore, the detection value is set to 2. 99.9% of the ratio values in unmarked images fall below T3, and the corresponding detection value is set to 3. The threshold values, T1, T2 and T3, may be determined by performing a statistical analysis of several images. The mapping of ratios to detection values based on the statistical distribution may be implemented in a look up table.

The statistical analysis may also include a maximum likelihood analysis. In such an analysis, an off-line detector generates detection value statistics for both marked and unmarked images. Based on the probability distributions of marked and unmarked images, it determines the likelihood that a given detection value for an input image originates from a marked and unmarked image.

At the end of these correlation stages, the detector has computed a ranked set of rotation-scale vectors 990, each with a quantized measure of correlation associated with it. At this point, the detector could simply choose the rotation and scale vectors with the highest rank and proceed to compute other orientation parameters, such as differential scale and translation. Instead, the detector gathers more evidence to refine the rotation-scale vector estimates. Fig. 15 is a flow diagram illustrating a process for refining the orientation

parameters using evidence of the orientation signal collected from blocks in the current frame.

Continuing in the current frame, the detector proceeds to compare the rotation and scale parameters from different blocks (e.g., block 0, block 1, block 2; 1000, 1002, and 1004 in Fig. 15). In a process referred to as interblock coincidence matching 1006, it looks for similarities between rotation-scale parameters that yielded the highest correlation in different blocks. To quantify this similarity, it computes the geometric distance between each peak in one block with every other peak in the other blocks. It then computes the probability that peaks will fall within this calculated distance. There are a variety of ways to calculate the probability. In one implementation, the detector computes the geometric distance between two peaks, computes the circular area encompassing the two peaks (π(geometric distance)^2), and computes the ratio of this area to the total area of the block. Finally, it quantizes this probability measure for each pair of peaks (1008) by computing the log (base 10) of the ratio of the total area over the area encompassing the two peaks. At this point, the detector has calculated two detection values: quantized peak value, and the quantized distance metric.

The detector now forms multi-block grouping of rotation-scale vectors and computes a combined detection value for each grouping (1010). The detector groups vectors based on their relative geometric proximity within their respective blocks. It then computes the combined detection value by combining the detection values of the vectors in the group (1012). One way to compute a combined detection value is to add the detection values or add a weighted combination of them.

Having calculated the combined detection values, the detector sorts each grouping by its combined detection value (1014). This process produces a set of the top groupings of unrefined rotation-scale candidates, ranked by detection value 1016. Next, the detector weeds out rotation-scale vectors that are not promising by excluding those groupings whose combined detection values are below a threshold (the "refine threshold" 1018). The detector then refines each individual rotation-scale vector candidate within the remaining groupings.

The detector refines a rotation-scale vector by adjusting the vector and checking to see whether the adjustment results in a better correlation. As noted above, the detector may simply pick the best rotation-scale vector based on the evidence collected thus far, and refine

only that vector. An alternative approach is to refine each of the top rotation-scale vector candidates, and continue to gather evidence for each candidate. In this approach, the detector loops over each vector candidate (1020), refining each one.

One approach of refining the orientation vector is as follows:

- fix the orientation signal impulse functions ("points") within a valid boundary (1022);
- pre-refine the rotation-scale vector (1024);
- find the major axis and re-fix the orientation points (1026); and
- refine each vector with the addition of a differential scale component (1028).

In this approach, the detector pre-refines a rotation-scale vector by incrementally adjusting one of the parameters (scale, rotation angle), adjusting the orientation points, and then summing a point-wise multiplication of the orientation pattern and the image block in the Fourier magnitude domain. The refiner compares the resulting measure of correlation with previous measures and continues to adjust one of the parameters so long as the correlation increases. After refining the scale and rotation angle parameters, the refiner finds the major axis, and re-fixes the orientation points. It then repeats the refining process with the introduction of differential scale parameters. At the end of this process, the refiner has converted each scale-rotation candidate to a refined 4D vector, including rotation, scale, and two differential scale parameters.

At this stage, the detector can pick a 4D vector or set of 4D vector and proceed to calculate the final remaining parameter, translation. Alternatively, the detector can collect additional evidence about the merits of each 4D vector.

One way to collect additional evidence about each 4D vector is to re-compute the detection value of each orientation vector candidate (1030). For example, the detector may quantize the correlation value associated with each 4D vector as described above for the rotation-scale vector peaks (see item 988, Fig. 14 and accompanying text). Another way to collect additional evidence is to repeat the coincidence matching process for the 4D vectors. For this coincidence matching process, the detector computes spatial domain vectors for each candidate (1032), determines the distance metric between candidates from different blocks,

and then groups candidates from different blocks based on the distance metrics (1034). The detector then re-sorts the groups according to their combined detection values (1036) to produce a set of the top P groupings 1038 for the frame.

Fig. 16 is a flow diagram illustrating a method for aggregating evidence of the orientation signal from multiple frames. In applications with multiple frames, the detector collects the same information for orientation vectors of the selected blocks in each frame (namely, the top P groupings of orientation vector candidates, e.g., 1050, 1052 and 1054). The detector then repeats coincidence matching between orientation vectors of different frames (1056). In particular, in this inter-frame mode, the detector quantizes the distance metrics computed between orientation vectors from blocks in different frames (1058). It then finds inter-frame groupings of orientation vectors (super-groups) using the same approach described above (1060), except that the orientation vectors are derived from blocks in different frames. After organizing orientation vectors into super-groups, the detector computes a combined detection value for each super-group (1062) and sorts the super-groups by this detection value (1064). The detector then evaluates whether to proceed to the next stage (1066), or repeat the above process of computing orientation vector candidates from another frame (1068).

If the detection values of one or more super-groups exceed a threshold, then the detector proceeds to the next stage. If not, the detector gathers evidence of the orientation signal from another frame and returns to the inter-frame coincidence matching process. Ultimately, when the detector finds sufficient evidence to proceed to the next stage, it selects the super-group with the highest combined detection value (1070), and sorts the blocks based on their corresponding detection values (1072) to produce a ranked set of blocks for the next stage (1074).

4.3 Estimating Translation Parameters

Fig. 17 is a flow diagram illustrating a method for estimating translation parameters of the orientation signal, using information gathered from the previous stages.

In this stage, the detector estimates translation parameters. These parameters indicate the starting point of a watermarked block in the spatial domain. The translation parameters, along with rotation, scale and differential scale, form a complete 6D orientation vector. The

6D vector enables the reader to extract luminance sample data in approximately the same orientation as in the original watermarked image.

One approach is to use generalized match filtering to find the translation parameters that provide the best correlation. Another approach is to continue to collect evidence about the orientation vector candidates, and provide a more comprehensive ranking of the orientation vectors based on all of the evidence gathered thus far. The following paragraphs describe an example of this type of an approach.

To extract translation parameters, the detector proceeds as follows. In the multi-frame case, the detector selects the frame that produced 4D orientation vectors with the highest detection values (1080). It then processes the blocks 1082 in that frame in the order of their detection value. For each block (1084), it applies the 4D vector to the luminance data to generate rectified block data (1086). The detector then performs dual axis filtering (1088) and the window function (1090) on the data. Next, it performs an FFT (1092) on the image data to generate an array of Fourier data. To make correlation operations more efficient, the detector buffers the fourier values at the orientation points (1094).

The detector applies a generalized match filter 1096 to correlate a phase specification of the orientation signal (1098) with the transformed block data. The result of this process is a 2D array of correlation values. The peaks in this array represent the translation parameters with the highest correlation. The detector selects the top peaks and then applies a median filter to determine the center of each of these peaks. The center of the peak has a corresponding correlation value and sub-pixel translation value. This process is one example of getting translation parameters by correlating the Fourier phase specification of the orientation signal and the image data. Other methods of phase locking the image data with a synchronization signal like the orientation signal may also be employed.

Depending on the implementation, the detector may have to resolve additional ambiguities, such as rotation angle and flip ambiguity. The degree of ambiguity in the rotation angle depends on the nature of the orientation signal. If the orientation signal is octally symmetric (symmetric about horizontal, vertical and diagonal axes in the spatial frequency domain), then the detector has to check each quadrant (0-90, 90-180, 180-270, and 270-360 degrees) to find out which one the rotation angle resides in. Similarly, if the

orientation signal is quad symmetric, then the detector has to check two cases, 0-180 and 180-270.

The flip ambiguity may exist in some applications where the watermarked image can be flipped. To check for rotation and flip ambiguities, the detector loops through each possible case, and performs the correlation operation for each one (1100).

At the conclusion of the correlation process, the detector has produced a set of the top translation parameters with associated correlation values for each block. To gather additional evidence, the detector groups similar translation parameters from different blocks (1102), calculates a group detection value for each set of translation parameters 1104, and then ranks the top translation groups based on their corresponding group detection values 1106.

4.4 Refining Translation Parameters

Having gathered translation parameter estimates, the detector proceeds to refine these estimates. Fig. 18 is a flow diagram illustrating a process for refining orientation parameters. At this stage, the detector process has gathered a set of the top translation parameter candidates 1120 for a given frame 1122. The translation parameters provide an estimate of a reference point that locates the watermark, including both the orientation and message components, in the image frame. In the implementation depicted here, the translation parameters are represented as horizontal and vertical offsets from a reference point in the image block from which they were computed.

Recall that the detector has grouped translation parameters from different blocks based on their geometric proximity to each other. Each pair of translation parameters in a group is associated with a block and a 4D vector (rotation, scale, and 2 differential scale parameters). As shown in Fig. 18, the detector can now proceed to loop through each group (1124), and through the blocks within each group (1126), to refine the orientation parameters associated with each member of the groups. Alternatively, a simpler version of the detector may evaluate only the group with the highest detection value, or only selected blocks within that group.

Regardless of the number of candidates to be evaluated, the process of refining a given orientation vector candidate may be implemented in a similar fashion. In the refining process, the detector uses a candidate orientation vector to define a mesh of sample blocks

for further analysis (1128). In one implementation, for example, the detector forms a mesh of 32 by 32 sample blocks centered around a seed block whose upper right corner is located at the vertical and horizontal offset specified by the candidate translation parameters. The detector reads samples from each block using the orientation vector to extract luminance samples that approximate the original orientation of the host image at encoding time.

The detector steps through each block of samples (1130). For each block, it sets the orientation vector (1132), and then uses the orientation vector to check the validity of the watermark signal in the sample block. It assesses the validity of the watermark signal by calculating a figure of merit for the block (1134). To further refine the orientation parameters associated with each sample block, the detector adjusts selected parameters (e.g., vertical and horizontal translation) and re-calculates the figure of merit. As depicted in the inner loop in Fig. 18 (block 1136 to 1132), the detector repeatedly adjusts the orientation vector and calculates the figure of merit in an attempt to find a refined orientation that yields a higher figure of merit.

The loop (1136) may be implemented by stepping through a predetermined sequence of adjustments to parameters of the orientation vectors (e.g., adding or subtracting small increments from the horizontal and vertical translation parameters). In this approach, the detector exits the loop after stepping through the sequence of adjustments. Upon exiting, the detector retains the orientation vector with the highest figure of merit.

There are a number of ways to calculate this figure of merit. One figure of merit is the degree of correlation between a known watermark signal attribute and a corresponding attribute in the signal suspected of having a watermark. Another figure of merit is the strength of the watermark signal (or one of its components) in the suspect signal. For example, a figure of merit may be based on a measure of the watermark message signal strength and/or orientation pattern signal strength in the signal, or in a part of the signal from which the detector extracts the orientation parameters. The detector may computes a figure of merit based the strength of the watermark signal in a sample block. It may also compute a figure of merit based on the percentage agreement between the known bits of the message and the message bits extracted from the sample block.

When the figure of merit is computed based on a portion of the suspect signal, the detector and reader can use the figure of merit to assess the accuracy of the watermark signal detected and read from that portion of the signal. This approach enables the detector to assess the merits of orientation parameters and to rank them based on their figure of merit. In addition, the reader can weight estimates of watermark message values based on the figure of merit to recover a message more reliably.

The process of calculating a figure of merit depends on attributes the watermark signal and how the embedder inserted it into the host signal. Consider an example where the watermark signal is added to the host signal. To calculate a figure of merit based on the strength of the orientation signal, the detector checks the value of each sample relative to its neighbors, and compares the result with the corresponding sample in a spatial domain version of the orientation signal. When a sample's value is greater than its neighbors, then one would expect that the corresponding orientation signal sample to be positive. Conversely, when the sample's value is less than its neighbors, then one would expect that the corresponding orientation sample to be negative. By comparing a sample's polarity relative to its neighbors with the corresponding orientation sample's polarity, the detector can assess the strength of the orientation signal in the sample block. In one implementation, the detector makes this polarity comparison twice for each sample in an N by N block (e.g., N = 32, 64, etc): once comparing each sample with its horizontally adjacent neighbors and then again comparing each sample with its vertically adjacent neighbors. The detector performs this analysis on samples in the mesh block after re-orienting the data to approximate the original orientation of the host image at encoding time. The result of this process is a number reflecting the portion of the total polarity comparisons that yield a match.

To calculate a figure of merit based on known signature bits in a message, the detector invokes the reader on the sample block, and provides the orientation vector to enable the reader to extract coded message bits from the sample block. The detector compares the extracted message bits with the known bits to determine the extent to which they match. The result of this process is a percentage agreement number reflecting the portion of the extracted message bits that match the known bits. Together the test for the orientation signal and the message signal provide a figure of merit for the block.

As depicted in the loop from blocks 1138 to 1130, the detector may repeat the process of refining the orientation vector for each sample block around the seed block. In this case, the detector exits the loop (1138) after analyzing each of the sample blocks in the mesh defined previously (1128). In addition, the detector may repeat the analysis in the loop through all blocks in a given group (1140), and in the loop through each group (1142).

After completing the analysis of the orientation vector candidates, the detector proceeds to compute a combined detection value for the various candidates by compiling the results of the figure of merit calculations. It then proceeds to invoke the reader on the orientation vector candidates in the order of their detection values.

4.5 Reading the watermark

Fig. 19 is a flow diagram illustrating a process for reading the watermark message. Given an orientation vector and the corresponding image data, the reader extracts the raw bits of a message from the image. The reader may accumulate evidence of the raw bit values from several different blocks. For example, in the process depicted in Fig. 19, the reader uses refined orientation vectors for each block, and accumulates evidence of the raw bit values extracted from the blocks associated with the refined orientation vectors.

The reading process begins with a set of promising orientation vector candidates 1150 gathered from the detector. In each group of orientation vector candidates, there is a set of orientation vectors, each corresponding to a block in a given frame. The detector invokes the reader for one or more orientation vector groups whose detection values exceed a predetermined threshold. For each such group, the detector loops over the blocks in the group (1152), and invokes the reader to extract evidence of the raw message bit values.

Recall that previous stages in the detector have refined orientation vectors to be used for the blocks of a group. When it invokes the reader, the detector provides the orientation vector as well as the image block data (1154). The reader scans samples starting from a location in a block specified by the translation parameters and using the other orientation parameters to approximate the original orientation of the image data (1156).

As described above, the embedder maps chips of the raw message bits to each of the luminance samples in the original host image. Each sample, therefore, may provide an estimate of a chip's value. The reader reconstructs the value of the chip by first predicting

the watermark signal in the sample from the value of the sample relative to its neighbors as described above (1158). If the deduced value appears valid, then the reader extracts the chip's value using the known value of the pseudo-random carrier signal for that sample and performing the inverse of the modulation function originally used to compute the watermark information signal (1160). In particular, the reader performs an exclusive OR operation on the deduced value and the known carrier signal bit to get an estimate of the raw bit value. This estimate serves as an estimate for the raw bit value. The reader accumulates these estimates for each raw bit value (1162).

As noted above, the reader computes an estimate of the watermark signal by predicting the original, un-watermarked signal and deriving an estimate of the watermark signal based on the predicted signal and the watermarked signal. It then computes an estimate of a raw bit value based on the value of the carrier signal, the assignment map that maps a raw bit to the host image, and the relationship among the carrier signal value, the raw bit value, and the watermark signal value. In short, the reader reverses the embedding functions that modulate the message with the carrier and apply the modulated carrier to the host signal. Using the predicted value of the original signal and an estimate of the watermark signal, the reader reverses the embedding functions to estimate a value of the raw bit.

The reader loops over the candidate orientation vectors and associated blocks, accumulating estimates for each raw bit value (1164). When the loop is complete, the reader calculates a final estimate value for each raw bit from the estimates compiled for it. It then performs the inverse of the error correction coding operation on the final raw bit values (1166). Next, it performs a CRC to determine whether the read is valid. If no errors are detected, the read operation is complete and the reader returns the message (1168).

However, if the read is invalid, then the detector may either attempt to refine the orientation vector data further, or start the detection process with a new frame. Preferably, the detector should proceed to refine the orientation vector data when the combined detection value of the top candidates indicates that the current data is likely to contain a strong watermark signal. In the process depicted in Fig. 19, for example, the detector selects a processing path based on the combined detection value (1170). The combined detection value may be calculated in a variety of ways. One approach is to compute a combined

detection value based on the geometric coincidence of the top orientation vector candidates and a compilation of their figures of merit. The figure of merit may be computed as detailed earlier.

For cases where the read is invalid, the processing paths for the process depicted in Fig. 19 include: 1) refine the top orientation vectors in the spatial domain (1172); 2) invoke the translation estimator on the frame with the next best orientation vector candidates (1174); and 3) re-start the detection process on a new frame (assuming an implementation where more than one frame is available)(1176). These paths are ranked in order from the highest detection value to the lowest. In the first case, the orientation vectors are the most promising. Thus, the detector re-invokes the reader on the same candidates after refining them in the spatial domain (1178). In the second case, the orientation vectors are less promising, yet the detection value indicates that it is still worthwhile to return to the translation estimation stage and continue from that point. Finally, in the final case, the detection value indicates that the watermark signal is not strong enough to warrant further refinement. In this case, the detector starts over with the next new frame of image data.

In each of the above cases, the detector continues to process the image data until it either makes a valid read, or has failed to make a valid read after repeated passes through the available image data.

5.0 Operating Environment for Computer Implementations

Figure 20 illustrates an example of a computer system that serves as an operating environment for software implementations of the watermarking systems described above. The embedder and detector implementations are implemented in C/C++ and are portable to many different computer systems. Fig. 20 generally depicts one such system.

The computer system shown in Fig. 20 includes a computer 1220, including a processing unit 1221, a system memory 1222, and a system bus 1223 that interconnects various system components including the system memory to the processing unit 1221.

The system bus may comprise any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using a bus architecture such as PCI, VESA, Microchannel (MCA), ISA and EISA, to name a few.

The system memory includes read only memory (ROM) 1224 and random access memory (RAM) 1225. A basic input/output system 1226 (BIOS), containing the basic routines that help to transfer information between elements within the computer 1220, such as during start-up, is stored in ROM 1224.

The computer 1220 further includes a hard disk drive 1227, a magnetic disk drive 1228, e.g., to read from or write to a removable disk 1229, and an optical disk drive 1230, e.g., for reading a CD-ROM or DVD disk 1231 or to read from or write to other optical media. The hard disk drive 1227, magnetic disk drive 1228, and optical disk drive 1230 are connected to the system bus 1223 by a hard disk drive interface 1232, a magnetic disk drive interface 1233, and an optical drive interface 1234, respectively. The drives and their associated computer-readable media provide nonvolatile storage of data, data structures, computer-executable instructions (program code such as dynamic link libraries, and executable files), etc. for the computer 1220.

Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and an optical disk, it can also include other types of media that are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, and the like.

A number of program modules may be stored in the drives and RAM 1225, including an operating system 1235, one or more application programs 1236, other program modules 1237, and program data 1238.

A user may enter commands and information into the computer 1220 through a keyboard 1240 and pointing device, such as a mouse 1242. Other input devices may include a microphone, joystick, game pad, satellite dish, digital camera, scanner, or the like. A digital camera or scanner 43 may be used to capture the target image for the detection process described above. The camera and scanner are each connected to the computer via a standard interface 44. Currently, there are digital cameras designed to interface with a Universal Serial Bus (USB), Peripheral Component Interconnect (PCI), and parallel port interface. Two emerging standard peripheral interfaces for cameras include USB2 and 1394 (also known as firewire and iLink).

Other input devices may be connected to the processing unit 1221 through a serial port interface 1246 or other port interfaces (e.g., a parallel port, game port or a universal serial bus (USB)) that are coupled to the system bus.

A monitor 1247 or other type of display device is also connected to the system bus 1223 via an interface, such as a video adapter 1248. In addition to the monitor, computers typically include other peripheral output devices (not shown), such as speakers and printers.

The computer 1220 operates in a networked environment using logical connections to one or more remote computers, such as a remote computer 1249. The remote computer 1249 may be a server, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the computer 1220, although only a memory storage device 1250 has been illustrated in Figure 20. The logical connections depicted in Figure 20 include a local area network (LAN) 1251 and a wide area network (WAN) 1252. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer 1220 is connected to the local network 1251 through a network interface or adapter 1253. When used in a WAN networking environment, the computer 1220 typically includes a modem 1254 or other means for establishing communications over the wide area network 1252, such as the Internet. The modem 1254, which may be internal or external, is connected to the system bus 1223 via the serial port interface 1246.

In a networked environment, program modules depicted relative to the computer 1220, or portions of them, may be stored in the remote memory storage device. The processes detailed above can be implemented in a distributed fashion, and as parallel processes. It will be appreciated that the network connections shown are exemplary and that other means of establishing a communications link between the computers may be used.

While the computer architecture depicted in Fig. 20 is similar to typical personal computer architectures, aspects of the invention may be implemented in other computer architectures, such as hand-held computing devices like Personal Digital Assistants, audio and/video players, network appliances, telephones, etc.

6.0 Color Adaptive Image Processing

In image processing applications, it is sometimes useful to be able to change the colors of an image while reducing the visibility of these changes. Image watermarking is one application where it is desirable to alter image samples to encode information in a manner that is readily recoverable by an automated process, yet substantially imperceptible to human visual perception. Often, the aim of watermark encoding is to maximize a watermark signal without significantly affecting image quality. Since the eye is more sensitive to changes in memory colors such as flesh tones or blue sky, it is beneficial to have a method of selectively controlling strength of a watermark in certain color regions. The following description proposes a framework for selective color masking of a watermark.

A watermark encodes auxiliary information in an image by making changes to image samples. A color masking framework maps a change in an image sample attribute to an equivalent yet less perceptible change in the color values of that image sample. This mapping can be used to obtain equal perceptual watermark changes to image samples in other areas of color space and to apply the change in the least visible color channels.

The mapping process has two primary inputs. One input to the mapping process is a desired change to an image attribute, which is dependent on the watermark. A second input is a set of color values of an image sample in the host image. Using these inputs, the mapping process computes a change in color components (or the final modified color value) of the host image.

Fig. 21 is a flow diagram illustrating a process for color masking. In this process, a desired change in an attribute (1300) of an image sample is mapped to changes in color components of the sample. A first stage (1302) of the process computes an adjustment for each color component based on the color values (1304) of an image sample. A second stage (1306) of the process takes the desired change in the image sample attribute (1300) and the adjustments for each color component, and determines a change in each color component necessary to effect the desired change in the image sample attribute. The process repeats for each of the image samples to be modified (1308).

The first and second stages can be merged into one computation that takes the input color vector of an image sample and change in an image sample attribute for that sample, and

computes changes in color components of the image sample For example, the change in color values can be computed analytically as a function of a change in an image sample attribute (like luminance or chrominance) and a color vector associated with the image sample.

The adjustment calculated in the first stage may be implemented as a set of scale factors that scale the change in the attribute of the image sample into corresponding changes to each of the color components of the image sample. The mapping of color components to a corresponding set of scale factors may be pre-computed and stored in a look-up table. For watermarking applications, this look up table is invoked in watermark encoding and, in some cases, decoding operations to determine a mapping between a change in an image sample attribute and corresponding changes in color components of the image sample.

Alternatively, the mapping of color components to a corresponding scale factor may be computed analytically as a function of the color components during encoding and decoding operations.

As noted, one way to map a change in an image sample attribute to color changes is to use a look up table (LUT). In a look up table, the inputs to the mapping process are used as an index to look up the change in color component. To illustrate a look up table for color mapping applications, consider the following examples. A 3D LUT that makes a one to one mapping of a luminance change to a color triplet is entirely comprised of triplet entries having values 1.0, 1.0, 1.0, which represent scaling factors that scale each color equally. Since for CMYK print images

$$\text{Luminance} = 0.3*C_1 + 0.6*M_1 + 0.1*Y_1$$

Where, $C_1M_1Y_1$ are the equivalent cyan, magenta and yellow ink values after black has been recombined. Thus a 3D LUT which does luminance scaling along the gray axis, and changes to a color scaling for saturated colors gradually changes from a 1.0, 1.0, 1.0 entry for gray image samples to a color scale value (for example 0.0, 0.0, 10.0 for a saturated yellow image sample). The value for the saturated color scaling for a given color triplet is calculated from

the desired luminance change of the watermark to give a similar luminance change but in a color which introduces the least visual artifacts.

Two considerations of the color masking framework are:

1. smooth transitions in and out of a color region;
2. a user interface for specifying a color region.

These considerations are explored further in implementations described below.

6.1 Evaluating Visibility of Changes to Image Sample Attributes

Since the effectiveness of a color mapping process is dependent on the human visual system, one begins by investigating the sensitivity of the human eye to various color changes.

Research in the field of broadcast color television has made the following observations about the sensitivity of human eyes to colors and color changes (See R.W.G. Hunt, 'The Reproduction of Color in Photography, Printing and Television', pages 380-383):

1. Blue saving in NTSC, since the eye is about a fifth less sensitive to high frequencies in blue-yellow than red-green.
2. The eye is much less sensitive to high frequency changes in chrominance than luminance. About 2.5 times the noise can be tolerated in the color signal as the luminance signal.

The first observation may be exploited to reduce watermark visibility by concentrating the watermark in the yellow/blue channel for printing. A 3D Look Up Table (LUT) may perform a mapping to the yellow/blue channel automatically, while changing to luminance scaling along the gray axis. The second item suggests changing the colors to move in a perpendicular direction to the gray axis, so that the watermark is a chrominance change. Once again a 3D LUT may perform this mapping automatically, while changing to luminance scaling along the gray axis.

The following sections describe two example watermark implementations for mapping changes in an image sample attributes to less visible colors. A first implementation takes a luminance change and color values of a host image and produces an equivalent luminance change to the host image in the colors less visible to the eye. Because the change to luminance is equivalent, the color mapping does not impact a watermark decoder that operates on luminance changes to extract the watermark.

A second implementation makes chrominance changes for certain colors. In such an implementation, the watermark decoder is designed to interpret chrominance changes correctly. The decoder should be implemented with a 3D LUT compatible with the watermark encoder so that it measures color changes in the correct direction in color space. For example, a pure chrominance change will produce no luminance change.

6.2 Mapping Luminance Changes

While the implementation details of watermark encoding schemes vary significantly, a class of watermarking schemes can be modeled as an array of changes to luminance values of a host image. The host image comprises an array of color vectors (e.g., an array of color triplets such as RGB, CMY, etc). The image sample may be represented as a vector between black and the pixel color value. To encode a watermark, the luminance of the image sample may be increased or decreased as shown in Fig. 22. Fig. 22 shows a 3 dimensional color space with Cyan (C), Magenta (M) and Yellow (Y) axes. The bold axis between black and white represents luminance. To make an equivalent luminance change in an image sample of a given color vector (C1, M1, Y1), one may make a corresponding scale to black as shown.

An alternative method of obtaining the same luminance change is to scale the image sample like a vector between white and the sample's color value as shown in Fig. 23. To make an equivalent luminance change, one may make a corresponding scale to white as shown.

For the same luminance change, the visibility changes as shown in the Table below.

Color	Scale to Black Colors Modified/Visibility	Scale to White Colors Modified/Visibility
yellow	cyan/magenta – high	yellow - low
red	cyan – high	magenta/yellow - low
green	magenta – medium	cyan/yellow - medium
Blue	Yellow – low	Cyan/magenta – high
Cyan	Magenta/yellow – low	Cyan – high
Magenta	Cyan/yellow – medium	Magenta - medium

By using the scale to white method for colors with high yellow content such as yellow, red and green, and scale to black for blue, cyan and magenta a lower visibility watermark can be encoded with the same detectability.

Scaling to white results in the watermark being applied mainly to the dominant colors, and scaling to black implies that the watermark is mainly in the secondary colors. The yellow content can be calculated using a color opponent calculation as in Battiato et al, except using CMY data after black has been recombined. See S.Battiato, D.Catalano, G.Gallo and R.Gennaro, 'Robust watermarking for images based on color manipulation', Third International Image Hiding Workshop, 1999.

$$YBsat = 2*Y_l - C_l - M_l$$

And YBsat controls the ratio of 'scale to white' (ScaleToWhite) to 'scale to black' (ScaleToBlack) watermark.

$$ChangeInLuminance = ScaleToBlack*(1-YBsat) + ScaleToWhite*YBsat$$

For print images, CMYK may be converted to an equivalent CMY color, which is used to index the 3D LUT.

The above color mapping may be implemented in a 3D LUT. The entries of the LUT

include a set of CMY color vectors representing a sampling of a color space. The number of color vector entries in the set can be minimized by making a representative sampling of the color space. In practice, several color vectors may correspond to a single color vector entry in the table. Ideally, a given color vector should be associated with the color vector entry that is closest to it in the color space. The correspondence between a color vector in an input image and a corresponding color vector entry may be established by hashing the input vector (e.g., truncating the least significant bit or bits) and then matching the hashed vector with a corresponding color vector entry in the table.

Once the color vector entries are established, each of the entries is associated with a set of scale factors. The set includes a scale factor for each color component. The specific color components in the implementation depend on the color format of the image. For example, images in an RGB format have scale factors for each of the R, G and B color components. Similarly, images in a CMY format have scale factors for each of the C, M and Y components of each table entry. The scale factors for each entry are derived by rewriting the above mathematical expression and solving for each color's scale factor as a function of the known color component values.

As an alternative to pre-computing the scale factors and storing them in a table, they may be computed during watermark encoding or decoding operations in a similar manner used to construct the table.

6.3 Mapping to Chrominance Changes

A second color masking method maps a change in an image sample attribute to a change in chrominance for selected colors. This color masking method may be used to map a change made to an image sample to encode a watermark to a chrominance change for image samples with colors that fall in a specified color region.

Before describing an example of this approach, it is helpful to consider an illustration showing the relationship between chrominance and luminance changes. A measure of color saturation is to drop a perpendicular from a host image sample C_1,M_1,Y_1 to the gray axis as shown in Fig. 24. C_0 is the minimum of (C_1,M_1,Y_1). The larger the distance CS, the greater the color saturation. The distance CS is used to change the mapping function from luminance

to chrominance.

For values of CS close to zero, this mapping process uses a luminance only mapping, and for pure colors (and substantially pure colors), it uses a chrominance only mapping. This is may be achieved with a mapping function of the form:

mapping = CS*chrominance + (1-CS)*luminance

An example of a chrominance mapping would be to change yellow only, except near gray (where a yellow only change would be very visible). A compatible decoder measures changes in the yellow channel (or blue for RGB detection), except near gray where it measures luminance changes. By matching color mapping in the encoder and decoder, a stronger watermark signal can be extracted from watermarked images.

The type of mapping described in this section may be implemented in a look up table, or in a mathematical function evaluated during encoding or decoding operations.

6.4 Application of Color Adaptive Methods to Image Processing in a Transform Domain

Color masking may also be used in applications where images are processed in a transform domain, such as a Fourier, Wavelet, or DCT domain, to name a few. In many of these applications, an image is transformed into an array of coefficients in a selected transform domain, the coefficients are modified, and converted back into a spatial domain. One such application is a transform domain watermark encoding process, where an image is transformed to a set of transform coefficients, and selected coefficients are modified to encode an auxiliary message (typically a binary message of one or more bits or symbols).

One way to implement color masking in a transform domain image processing operation is to compute a characteristic color of an image region to be transformed into another domain and alter color components of the image region based on that characteristic color. As in the above methods, the characteristic color of an image region may be used to look up the color channels to be modified to effect the desired changes to the image. In a transform domain watermark encoding process, the changes made to transform coefficients

to encode a watermark can be targeted to specific color channels for each image block. For example, color channels to which the watermark is applied are altered depending on a characteristic color of an image block to be transformed to transform coefficients for watermark encoding. The characteristic color may be computed as an average, or DC component of the color for that block.

For watermark decoding, the characteristic color of an image block may be used to look up a corresponding color channel or channels in which a watermark is expected to be encoded.

To illustrate color adaptive watermarking in a transform domain, consider the following example method. The encoding method computes a characteristic color for a block of image samples. It uses this characteristic color to look up the color channels in which to embed the watermark. The calculation of the color channels may be implemented in a color look up table that maps in input color to a corresponding color channel or channels. Preferably, the watermark should be embedded in color channels that are the least visible for the given characteristic color.

After determining the color channels in which to encode the watermark, the encoder converts image samples of the block into those color channels. Next, the encoder transforms the block into a transform domain, alters the transform domain coefficients to encode an auxiliary signal, and performs an inverse transform back into the block's original domain. The encoded block may be transformed into a desired color format for rendering or output.

The watermark decoder computes the characteristic color for the block and looks up the color channel into which the watermark is expected to be encoded. It then proceeds to decode the watermark from those color channels.

6.5 A User Interface for Controlling Watermark Encoding in Color Regions

Adaptive color masking can be further enhanced by empowering the user to control how the watermark is applied to colors. This functionality can be implemented by making a user interface that allows the user to specify the intensity or strength of the watermark for specific colors or color regions. A particularly useful way to implement this functionality is to create an interactive user interface program that enables the user to control watermark strength in designated color regions and then immediately view the impact of the selected

strength on the image. Using this type of interactive interface, the user can see how color specific changes to watermark strength impact visual quality immediately.

One way to allow the user to specify watermark strength for selected colors is to allow the user to specify a color region and an associated color specific watermark strength parameter. Human eyes are sensitive to abrupt changes in color regions. As such, the watermark strength parameter should be applied in a manner that makes smooth transitions of watermark strength to and from a designated color region. For example, a color region may be defined by a start point C_1,M_1,Y_1, end point C_2,M_2,Y_2 and a Gaussian half-width W_g as shown in Fig. 25. The volume is like a fuzzy football in color space, which is defined by the user as explained below.

To illustrate an example user interface, consider the following implementation. The image to be watermarked is displayed in an encoder application context (e.g., in the Adobe Photoshop image editing program with a watermark encoder and decoder plug-in program). Within the user interface of the image editing program, the a user selects 'Define Color Region' from within the 'Embed Watermark' box. The user moves the cursor over the region to be protected, clicking at the points to sample the pixel colors while the user interface displays visual feedback of the color region selected. A default Gaussian half-width W_g may be used unless it is changed.

The strength of the watermark in this color region can then be set independently of the overall watermark strength. The suggested default value would be say 50% of the overall watermark strength, if 50% of the image is effected. The default would:

a) reduce, if a smaller percentage of the image was selected – for example (50-20)% of overall watermark strength if 30% of image selected.
b) increase, if a larger percentage of the image was selected – for example (50+20)% of overall watermark strength if 70% of image selected.

In addition several default color regions (red, green, blue, cyan, magenta, yellow, white, black and neutral), selective color retouch would be selectable. The watermark strength could be independently set for each color region. All settings can be saved to, or restored from a file.

Clearly, this example represents only one possible implementation. There are many other ways to allow the user to specify watermark strength for designated colors.

6.6 Using Color Mapping as a Key

Any of the above methods are image dependent because the watermark is encoded into color channels that depend on the specific color values of the image samples in the image. The color mapping could be used as a color key to determine the validity of a watermark. One way to check the validity is to check the color channels that contain a known watermark and compare these color channels with those expected to contain the watermark based on the color mapping key. Another way is to measure a figure of merit of a known watermark or watermark message payload in the color channels specified by the color mapping key. If the figure of merit falls below a threshold level, the detector can reject the watermark as being invalid, or reject the signal as being unmarked.

The term, "known watermark," refers to a watermark expected to be in an image suspected of containing a watermark based on knowledge of the watermark encoder. One example of such a watermark is the watermark orientation signal described above. Another example is a fixed watermark message signal that is common to all images that employ a similar watermark encoding process. To determine whether a watermark is valid, the detector calculates a figure of merit as described previously for the watermark in the color channels where the watermark is expected to be encoded based on the colors of the watermarked image. One such figure of merit is to measure the strength of a watermark orientation signal as described previously. Another figure of merit is the percentage agreement calculation described previously. If the figure of merit does not surpass a set threshold, the detector deems the watermark to be invalid, or the image to be unmarked.

The color mapping process, such as those described above, indicate the color region where the watermark will be the strongest based on the color values of the image samples in the suspect image. For example, using the implementation described above:

In red areas, the watermark should be encoded more strongly in magenta and yellow colors.

In blue areas, the watermark should be encoded more strongly in the yellow color.

The 3D LUT in effect acts as a key to which colors should be modified. This could

be used to help counter certain types of attacks, since the attacker would not have the color key. For example, this could be used where one attempts to forge a watermark without the color key or where one attempts to copy a watermark from a different image. Since the watermark is dependent on the color of the image, it will not be valid when copied from a different image. In particular, it will likely not be encoded in the correct color channels based on the colors of the image to which it is copied. As a result, the figure of merit measured for the watermark in these color channels is likely to indicate that the watermark is invalid.

7.0 Concluding Remarks

Having described and illustrated the principles of the technology with reference to specific implementations, it will be recognized that the technology can be implemented in many other, different, forms. For example, adaptive color masking can be applied in a variety of types of watermarking systems. It can also be applied in image and video coding applications to reduce visibility of artifacts due to lossy compression techniques. To provide a comprehensive disclosure without unduly lengthening the specification, applicants incorporate by reference the patents and patent applications referenced above.

The particular combinations of elements and features in the above-detailed embodiments are exemplary only; the interchanging and substitution of these teachings with other teachings in this and the incorporated-by-reference patents/applications are also contemplated.